Mining and Validation of Localized Frequent Web Access Patterns with Dynamic Tolerance

Olfa Nasraoui[1] and Suchandra Goswami[2]

[1] Dept. of Computer Engineering & Computer Science, University of Louisville
Louisville, KY 40292
olfa.nasraoui@louisville.edu
http://www.louisville.edu/~o0nasr01/
[2] Dept. of Electrical and Computer Engineering, The University of Memphis
Memphis, TN 38152-318
sgoswami@memphis.edu

Abstract. Mining user profiles is a crucial task for Web usage mining, and can be accomplished by mining frequent patterns. However, in the Web usage domain, sessions tend to be very sparse, and mining the right user profiles tends to be difficult. Either too few or too many profiles tend to be mined, partly because of problems in fixing support thresholds and intolerant matching. Also, in the Web usage mining domain, there is often a need for post-processing and validation of the results of mining. In this paper, we use criterion guided optimization to mine localized and error-tolerant transaction patterns, instead of using exact counting based method, and explore the effect of different post-processing options on their quality. Experiments with real Web transaction data are presented.

1 Introduction

Association rule mining [2] is a key data mining task that is most widely applied to problems in market basket analysis, where one desires to find baskets of items frequently purchased *together* in a Transactional Database (TDB). More recent applications include text mining [9], scientific applications [6], and bioinformatics [5]. Mining *frequent itemsets* or *Frequent Pattern* (*FP*) is a crucial prerequisite step to mining association rules that faces several previously acknowledged challenges, such as mining and maintaining association rules in very large databases [3,10,13] and in evolving databases [4,11]. In this paper we will focus our attention on three *other* issues that directly address the FP definition, in particular in the context of TDBs with a *large number of items/dimensionality*, *sparse* data, and *heterogenous* data distributions. Such databases are very common in the context of *e-commerce transactions* on large e-commerce sites that offer a huge number of products. Such TDBs are also very common in the context of mining of web *user clickstream data*, where user sessions are the transactions and URLs are the items. Other kinds of data that satisfy these characteristics occur in the context of mining large collections of *text documents*, where the documents play the role of transactions and keywords play the role of items. *Sparse* data sets suffer from the fact that for a large number of items, the number of non-zero entries is a very small fraction of the total number of entries in the transaction matrix.

X. Li, O.R. Zaiane, and Z. Li (Eds.): ADMA 2006, LNAI 4093, pp. 80 – 91, 2006.

For the majority of Web usage sessions or Web transactions data, frequent itemset definition and mining have suffered from the following problems:

(i) Sensitivity to Support Thresholds: Very low support thresholds typically lead to generating too many spurious patterns (that are due to random correlations in data), while high support thresholds risk missing many interesting patterns that occur with low support, but have high confidence. This problem most particularly affects heterogenous data sets, where certain itemsets may occur on only part of the data (e.g. in only some segments of a customer database depending on the geographical location), and hence will have low support. One of the first researchers who have addressed this issue are Pei, Tung, and Han [8] who defined the notion of *Fault-Tolerant Frequent Patterns* (*FTFP*). Unlike frequent itemsets, FTFPs allow a fault tolerance equal to δ, meaning that *up to* δ *mismatches* in the items are allowed. So instead of finding exact patterns in data the search is for approximate and more general fault-tolerant patterns. Unfortunately, this approach actually requires *two* instead of *one* support threshold: one for the items, and another one for the itemsets. Moreover, an *additional* threshold is required for the amount of tolerance, δ.

(ii) Error intolerance: When counting the support of an itemset, only transactions that completely include *all* the items of an itemset are counted. If a transaction matches an itemset in say 10 items, but fails to match *one* item, then it is completely excluded from the support count. In other words, this is an *all or nothing* counting. Some work has addressed this issue, including Fault-Tolerant Frequent Pattern (*FTFP*) [8], and Error-tolerant Itemsets (*ETI*) [12] of two types (*strong* and *weak*). The problem with this approach is that getting away from having to pre-specify support thresholds led to getting trapped in another requirement: having to specify tolerance thresholds.

(iii) Absence of locality in frequent itemset counting and validation: Data may be skewed, heterogeneous, or better modeled by several subsets, each with its own frequent itemsets, possibly with *different support* thresholds and even different levels of *tolerance*. Researchers who have addressed this issue include Aggarwal, Procopiuc, and Yu [1] who have proposed CLASD (**CL**ustering for **AS**sociation **D**iscovery) that discovers frequent patterns called *metatransactions* by aggregating the item frequencies of transactions assigned to each clusters based on maximum similarity into a frequency vector. The clusters are found using a Hierarchical Agglomerative Clustering that starts with randomly selected transactions as the initial seeds. While this approach proposed some improvements mainly in localization of the search, it still needs to address several problems: *(i)* sensitivity to prespecified number of clusters (k) and minimum size threshold of a cluster that implicitly plays the role of metatransaction support, *(ii)* also, since transactions are assigned based only on similarity, transactions that are not very similar to any cluster will still get lumped to the closest cluster, and hence contribute to its support. In other words, the notion of *cluster size* which is equivalent to *metatransaction support* does not take into account the *level* of similarity of the transaction. This corresponds to the *opposite* extreme end of exact/intolerant support counting because even a transaction that does not match any of the items in the candidate pattern will be counted in the support.

(iv) Post-processing and validation: In the Web usage mining domain, there is often a need for post-processing and validation of the results of frequent pattern mining, be-

cause these patterns are to be used downstream for specialized applications such as personalization. Thus, it is interesting to study how post-processing affects the results.

In this paper, we apply an FP mining approach that we have proposed in [14] to mining *local, error-tolerant* frequent patterns (profiles) from Web transaction data, and explore the effect of various post-processing options. We call the special kind of patterns that we mine: ***L**ocalized **E**rror **T**olerant **F**requent **P**attern* (*LET-FP).* Unlike the preliminary work in [14], in this paper, we will also explore several post-processing options of the mined frequent patterns, and extended our proposed information retrieval inspired validation procedure that simultaneously penalizes against *(i)* an excess of spurious patterns, and *(ii)* a lack of accurate patterns, by calculating measures that attempt to answer the following crucial questions: (1) Are all the mined patterns *needed* to summarize the data? (2) Is the data set well summarized/represented by the mined patterns?

2 A Generalized Frameworks for Localized Error-Tolerant Frequent Pattern (LET-FP) Mining

2.1 Frequent Itemsets: A Similarity Based Perspective

Frequent itemsets or patterns (FP) can be considered as one way to form a *summary* of the input data. As a summary, frequent patterns represent a reduced form of the data that is at the same time, *as close as possible* to the original input data. This is compatible with the notion of support as a critical measure of goodness for a FP. *Classical support* measures the *count* of the transactions that *completely include* a FP. Therefore transactions that are very similar to a FP, but perhaps lacking a single item from the FP do not even count in its support. The first step toward including *tolerance* is to allow transactions that are very similar to a FP to count in what we refer to as *partial* support. For this reason, we need to consider using a *similarity* measure to capture *closeness* between a FP and a transaction. We first explain the notation that will be used throughout the rest of this section. Hence, P_i denotes the i^{th} frequent pattern. $|P_i|$ is the number of items in P_i. t_j denotes the j^{th} transaction, $|t_j|$ is the number of items in t_j, and S_{ij} is the Similarity between the i^{th} FP and the j^{th} transaction. An FP should represent a *frequently* occurring trend. Hence it should be as *similar* as possible to *as many* transactions as possible. Hence, we need to assess the similarity between a FP, P_i, and each transaction t_j. For example it can be shown that the classical itemset definition of *APriori* uses a complete FP inclusion based similarity [14], where the similarity is nonzero (and has value 1) only if the pattern is completely included in a transaction. Other possibilities (that are investigated in this paper) include the *cosine* similarity, *precision*, *coverage*, as well as the *minimum of precision and coverage* (*MinPC*) measures of a candidate pattern, using the transaction as ground-truth reference [14].

2.2 Error Tolerant Support

Let a candidate ***L**ocalized **E**rror **T**olerant **F**requent **P**attern*, henceforth referred to as *LET-FP*, be denoted as P_i, and let the transactions in a DB be denoted by t_j. Instead of *Apriori's* complete pattern inclusion based matching, we propose to use a generalized,

softer matching measure. This matching measure $Sim(P_i,t_j)$ quantifies how faithfully the frequent pattern P_i serves as a summary for transaction t_j. A dissimilarity $d(P_i,t_j)$ can be defined so that it is inversely related to $Sim(P_i, t_j)$, for example,

$$d(P_i,t_j) = (1- Sim(P_i,t_j))^2 \tag{9}$$

Let the amount of tolerance ε be dynamic and defined in the same units as $d(P_i,t_j)$. Furthermore, let the tolerance be *localized*, and hence depend on the ETFP itself, i.e

$$\varepsilon_i = \varepsilon(P_i).$$

Next, let a *tolerance-normalized* dissimilarity between LET-FP P_i and transaction t_j be defined as

$$d_\varepsilon(P_i, t_j,\varepsilon_i) = \frac{d(P_i,t_j)}{\varepsilon_i} \tag{10}$$

A lower tolerance will tend to inflate the effect of dissimilarity, hence reflecting a more stringent matching process. Now that the tolerance degree, ε_i has been "absorbed" into the normalized dissimilarity $d_\varepsilon(P_i, t_j,\varepsilon_i)$, a measure of *support* that is *error-tolerant* can be defined. Let this localized error-tolerant support be defined as

$$s(P_i,t_j,\varepsilon_i) = f(d_\varepsilon(P_i,t_j,\varepsilon_i)) = f(d(P_i,t_j), \varepsilon_i),$$

where f: $\Re \rightarrow [0,1]$ is a monotonically non-increasing function. The *Total Localized Error-Tolerant support* of LET-FP P_i may be defined by summing the contributions from all transactions as follows

$$Ts(P_i,\varepsilon_i) = \sum_{t_j \in T} s(P_i,t_j,\varepsilon_i) \tag{11}$$

Note that the total support in (11) increases monotonically with the tolerance ε_i. Because tolerance is not known in advance, this will favor higher tolerance values. To put a limit on this bias, we define a "*normalized* support" instead of an absolute support, to be used as an FP goodness criterion, i.e,

$$\rho(P_i,\varepsilon_i) = \frac{Ts(P_i,\varepsilon_i)}{\varepsilon_i} \tag{12}$$

In this equation the tolerance degree can also be considered as a penalty factor P_i. Given the same support, LET-FPs will be rewarded if their tolerance degree is smaller and penalized if their tolerance-degree is higher.

2.3 Avoiding Fixed Support Thresholds: Mining Frequent Patterns by Support Optimization

Instead of searching for the FPs that exceeds a *fixed* support threshold, we propose to seek the FPs that *maximize the error tolerant support* in (12). The FP mining and tolerance search problem can be stated as an *alternating optimization problem* that boils down to two optimization steps, to determine the frequent *patterns* and the error *tolerance* respectively that optimize the error tolerant support: that is **(1)** Fix ε_i, and solve for $P_i = Arg\ Max\ (\rho(\boldsymbol{P_i},\varepsilon_i))$, and **(2)** Fix P_i, and solve for $\varepsilon_i = Arg\ Max\ (\rho(P_i,\boldsymbol{\varepsilon_i}))$.

Step 2 can be solved by *analytical* optimization if $\rho(P_i,\varepsilon_i)$ is *differentiable* with respect to ε_i, and a closed *Piccard update equation* for ε_i can be derived as shown in [14]. The normalized error-tolerant support in (12) satisfies several desiderata.

- Localized Support: Support is defined on increasingly smaller subsets/clusters of the data, providing a *localized* and confined counting.
- Error-tolerance: Data tuples that deviate slightly from candidate LET-FP will still contribute to its support, though to a lesser degree, depending on the tolerance amount.
- Dynamic Tolerance: Given the local support measure function $s(P_i,t_j,\varepsilon_i) = f(d_\varepsilon(P_i,t_j,\varepsilon_i)) = e^{-d_\varepsilon(P_i,t_j,\varepsilon_i)}$, we can analytically derive an iterative update equation [14] for dynamic tolerance level ε_i based on optimizing the total error-tolerant support given by (12). However $\rho(P_i,\varepsilon_i)$ is not in a form that is differentiable with respect to P_i. Therefore, a *non-analytical* optimization approach is needed for Step 1. We use a Genetic algorithm for this purpose, but do not rule out other heuristic optimization methods. This leads to an *alternating optimization* approach, i.e. alternate solving for one of the parameters, while the rest are fixed, and it is common in the optimization and machine learning literature, including the *Expectation Maximization (EM)* algorithm, and *Maximum Likelihood Estimation* methods.

2.4 Localized ETFP Mining by Partitioning and Zooming

Some strong (i.e. highly accurate/confident) associations may lurk in small segments of a huge data set. In this case, they risk being missed because of their low support. In other words, finding frequent itemsets from the entire aggregate data may not be able to reveal patterns that are only valid in *small localized* segments of the data. Data *locality* concepts can offer several advantages in this context. We achieve locality by gradually focusing the search on smaller and smaller segments of the data set. A greedy procedure extracts the unique optimal patterns discovered at each iteration. Redundant patterns are identified based on their compatibility with a previously extracted pattern, and are therefore ignored.

Based on these extracted FPs, the dataset is gradually divided into smaller parts/clusters containing similar transactions. The steps needed to obtain localized LET-FPs may be summarized as follows:

Algorithm LET-FP Mining:

0. Let *current* transaction dataset (D_c) = D (input data), and let the set of final extracted LET-FPs, $P=\varnothing$
1. Initial FP-Generation: Seed the FPs by selecting random samples from *current* transaction dataset (D_c).
2. Iterative LET-FP search and extraction: will result in C LET-FPs $P_1,\ldots,P_c$, and tolerance values $\varepsilon_1,\ldots,\varepsilon_c$
3. Partition transactions by assigning each transaction to nearest LET-FP (based on chosen similarity measure). This will partition the dataset into C subsets $T_1,\ldots,T_c$.
4. For $i = 1, \ldots, C$ { **// Step 4 is for zooming (optional)**
   Let $T_i^{out-of-core} = \{t_j \mid s(P_i,t_j,\varepsilon_i) < s_{core}\}$
   Let $T_i = T_i - T_i^{out-of-core}$

```
      Let T_Zoom = ⋃_i T_i^(out-of-core)
    }
5. For each subset T_i {
     If ε_i>ε_max and |T_i| > t_max Then {
       Let current transaction dataset D_c = T_i .
       Go to step 1,       // repeat search on each cluster
     }
     Else P=P∪P_i          // Add to final list of LET-FPs
  }
6. Let current dataset D_c = T_Zoom. Go to Step 1. // Step 6 is for zooming
   (optional)
```

Step 2 can be any competent search method, preferably, one that is global, and that can benefit from randomized search to sample the huge search space of all possible LET-FPs, such as a genetic algorithm.

2.5 Validation in an Information Retrieval Context

Frequent itemsets or patterns (FP) can be considered as one way to form a summary of the input data. As a summary, frequent patterns represent a *reduced* form of the data that is at the same time, *as close as possible* to the original input data. This description is reminiscent of an *information retrieval* scenario, in the sense that patterns that are retrieved should be as *close* as possible to the original transaction data. Closeness should take into account both *(i) precision* (a summary FP's items are all correct or included in the original input data, i.e. they include *only* the true data items) and *(ii) coverage/recall* (a summary FP's items are complete compared to the data that is summarized, i.e. they include *all* the data items). We propose a validation procedure that is inspired by information retrieval that simultaneously *penalizes (i) an excess of spurious patterns*, and *(ii)* a *lack* of *accurate* patterns, by calculating measures that attempt to answer the following crucial questions: **(1)** Are *all* the mined patterns *needed* to *(a) completely* and *(b) faithfully* summarize the data? **(2)** Is the data set *(a) completely* and *(b) faithfully* summarized/represented by the mined patterns? Each of these two questions is answered by computing *coverage/recall* as an *Interestingness measure to answer part (a),* and *precision as an Interestingness measure to answer part (b)*, while reversing the roles played by the mined patterns and the data transactions, respectively.

First, we compute the following *Interestingness* measures for each LET-FP, letting the *Interestingness measure*, $Int_{ij} = Cov_{ij}$ *(i.e., coverage:* See Eqs. In Sec 3.1.) *to answer part (a),* and $Int_{ij} = Prec_{ij}$ *(i.e., precision:* See Eqs. In Sec 3.1.) *to answer part (b).*

Let the set of transactions satisfying interestingness measure Int_{ij} for the i^{th} LET-FP, be

$$T_i^{\text{int}} = \{t_j \mid Int_{ij} > Int_{min}\}.$$

Then the following measure gives the proportion of transactions that are well summarized by the i^{th} LET-FP

$$Int_i^1 = \left|T_i^{\text{int}}\right| / |\mathrm{T}|$$

The average Interestingness over the entire set P of patterns is given by

$$Int^1 = \sum_i Int_i^1 / |P| \tag{16}$$

The measure in (16) is penalized if there are *too many patterns* that do not satisfy the interestingness criterion *for many transactions*. Hence, this is a very severe validation measure. When $Int_{ij} = Cov_{ij}$, we call Int^1 the *Normalized Count of Coverage*, and it answers Question 1.a. When $Int_{ij} = Prec_{ij}$, we call Int^1 the *Normalized Count of Precision*, and it answers Question 1.b.

Now, if we let $T^* = \{t_j \mid \underset{i}{Max}\ (Int_{ij}) > Int_{min}\}$. Then

$$Int^2 = |T^*| / |T| \tag{17}$$

When $Int_{ij} = Cov_{ij}$, we call Int^2 the *Cumulative Coverage of Transactions*, and it answers Question 2.a. When $Int_{ij} = Prec_{ij}$, we call Int^2 the *Cumulative Precision of Transactions*, and it answers Question 2.b.

The measures answering questions 2 quantify the quality of mined patterns from the point of view of providing an accurate summary of the input data. While the measures answering questions 1 quantify the necessity of any pattern at all, hence penalizing against patterns that are spurious, or for which there is no counterpart in the input data. The above measures are computed over the entire range of the Interestingness threshold Int_{min} from 0% to 100% in increments of 10%, and plotted.

2.6 LET-FP Search and Post-processing Options

After the completion of the LET-FP search algorithm, we partition the input transactions into as many clusters as the number, say $|P|$, of the *original* (i.e., *without post-processing*) LET-FPs, $P=\{P_1,\ldots, P_{|P|}\}$. Let these transaction clusters be denoted as $T_1,\ldots, T_{|P|}$. Then, there are several ways that we may use the LET-FPs, as listed below.

Search Options: First the search for LET-FPs can either use *zooming* or not.

(1) ***Standard*** **LET-FPs:** obtained by eliminating steps 4 and 6 in Algorithm LET-FP Mining (see Sec 3.4) and doing *no* post-processing
(2) ***Zoomed*** **LET-FPs:** We use steps 4 and 6 in Algorithm LET-FP Mining (see Sec 3.4) to gradually zoom into each transaction cluster by peeling off the out-of-core transactions.

Post-Processing Options: After completing the search for LET-FPs, there are several options:

(1) ***Original*** **LET-FPs:** These are the LET-FPs obtained *without* post-processing
(2) ***Aggregate*** **LET-FPs:** Frequency vectors computed by averaging the item occurrence frequencies in each cluster separately, then converting to a binary vector (1 if frequency > 0.10, 0 otherwise).
(3) ***Robustified*** **LET-FPs:** Before aggregating the LET-FP frequencies as in the previous option, we zoom into each cluster, and remove the out-of-core transactions, i.e, $T_i = T_i - \{t_j \mid s(P_i,t_j,\mathcal{E}_i) < s_{core}\}$.

Other options are produced by different combinations of search and post-processing options. Table 1 lists the different codes used in our experimental section that designate these different options

Table 1. Category codes corresponding to LET-FP search and post-processing options

Code	search	post-processing
spa	**S**tandard (i.e., no zooming)	**p**ost-processing**: a**ggregate
spr	**S**tandard (i.e., no zooming)	**p**ost-processing**: r**obustified
so	**S**tandard (i.e., no zooming)	**O**riginal (no post-processing)
zpa	**Z**oomed (w/ steps 4 & 6 of LET-FP Mining Algorithm)	**p**ost-processing**: a**ggregate
zpr	**Z**oomed (w/ steps 4 & 6 of LET-FP Mining Algorithm)	**p**ost-processing**: r**obustified
zo	**Z**oomed (w/ steps 4 & 6 of LET-FP Mining Algorithm)	**O**riginal (no post-processing)

3 Experimental Results

Experimental results are obtained by using the LET-FP Mining Algorithm described in Sec. 3.4, and we compare against the performance of *APriori* [2] with varying minimum support levels. The proposed LET-FP Mining algorithm is validated using the different *search* strategies (with or without zooming) and different *similarity* measures. Hence, we validate all the possible combinations listed in Sec. 3.7, as well as the different similarity measure options by computing and plotting the interestingness measures described in Sec. 3.6. To avoid information overload, for each <search & post-processing> category, we report the results *only for the best performing similarity measure*. Also because of the definition of frequent itemsets in *APriori*, precision is always equal to 1. Hence we do not plot it for different minimum interestingness thresholds, but rather list it in tabular format, considering it as threshold of 0.9. To optimize step 1, we use a randomization based optimization method (GA) with population size 100, 30 generations, and binary encoding of transactions. The crossover and mutation rates were 0.9 and 0.001 respectively. The dataset consists of the preprocessed web-log data of a Computer science department website. This dataset has 343 distinct URLs accessed by users. A session was defined as a sequence of URL requests from the same IP address within a prespecified time threshold of 45 minutes [7], leading to 1704 real user sessions. On this data set, *APriori* [2] generated a large number of itemsets, despite the conservative minimum support thresholds, as shown in Table 2, while the proposed LET-FP mining algorithm produced a much smaller number of frequent patterns as shown in Table 3 for the different search strategies (with or without zooming) and for different similarity measures (cosine, precision, coverage, MinPC).

Table 2. Variation in number of *APriori* itemsets with respect to support threshold (Web transaction data)

Support	1-itemsets	2-itemsets	3-itemsets	4-itemsets	5-itemsets	Total
1%	129	159	160	81	2	531
2%	87	40	26	18	-	171
5%	27	12	6	-	-	45

Table 3. Variation in number of LET-FPs for different similarity measures (web transaction data)

	Cosine	Coverage	Precision	MinPC
standard	36	19	31	38
zoomed	32	22	26	30

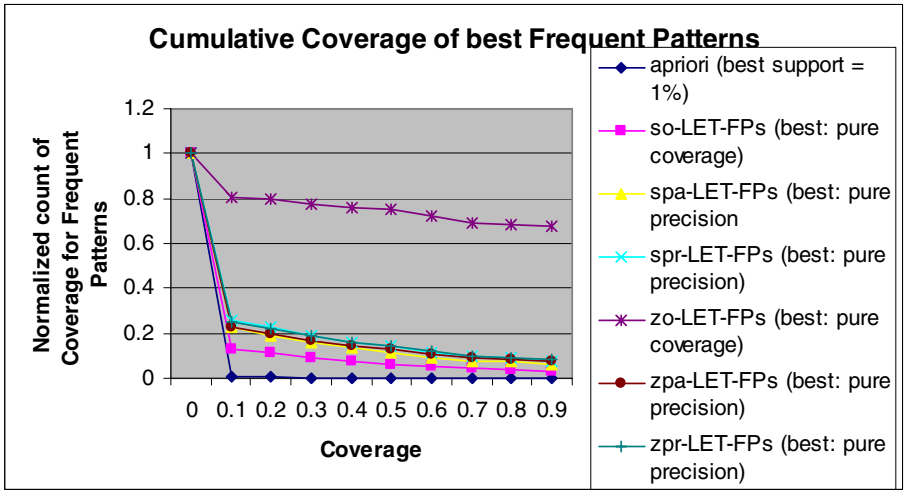

Fig. 1. Normalized-count of Coverage for best Frequent Patterns

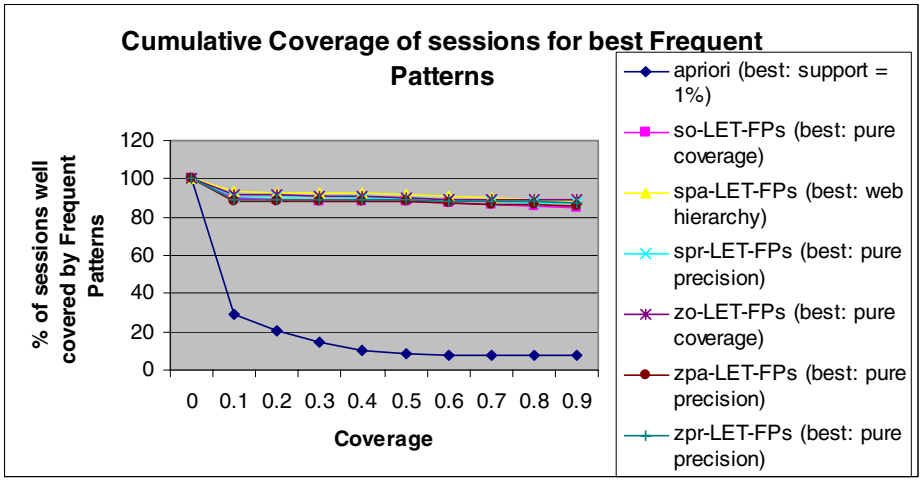

Fig. 2. Cumulative Coverage of sessions for best performing similarity in each category

Figure 1 shows (for all quality thresholds starting from 0 until 0.9) the normalized-count of coverage for FPs using the best similarity measure for LET-FPs and the best support percentage for *Apriori*. We can see that the LET-FPs perform better than the Frequent Patterns obtained by *Apriori*. Figure 2 shows that the percentage of sessions well covered is higher for LET-FPs than with *Apriori*.

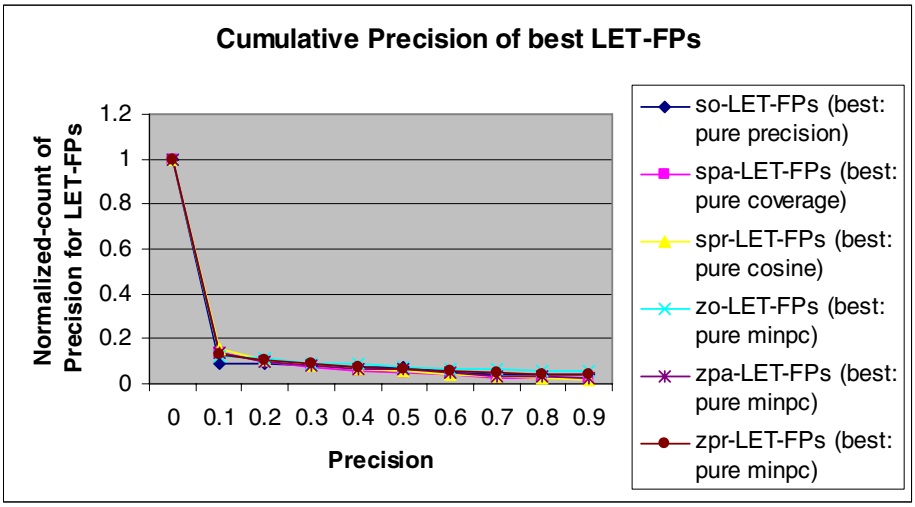

Fig. 3. Normalized-count of Precision for best performing similarity in each category

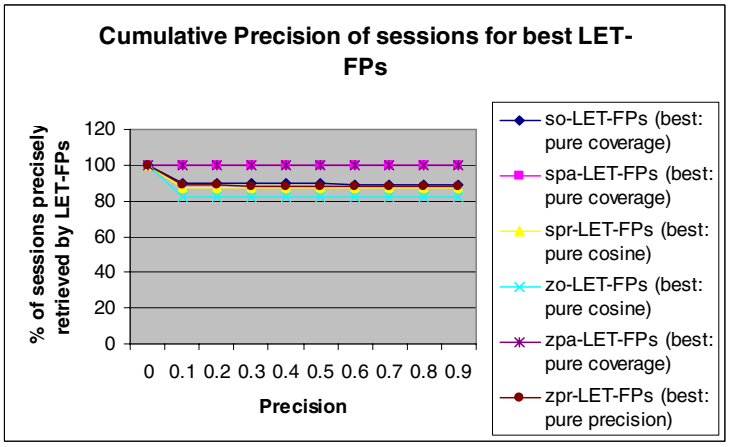

Fig. 4. Cumulative Precision of sessions for best performing similarity in each category

Figure 3 shows that *spr-LET-FPs* (i.e. *standard* search, *post-processed* with *robustification*) using pure cosine similarity give the best normalized-count of precision. Figure 4 shows that the *zpa-LET-FPs* using coverage similarity measure give the best

percentage of sessions precisely retrieved by FPs. To summarize all the results, we note that both normalized-counts of coverage and precision of FPs obtained by *Apriori* are less than those of LET-FPs. At the same time, the % of sessions/transactions well covered by any of the FPs obtained by *Apriori* is less than the % of sessions/transactions well covered by any of the LET-FPs obtained by our proposed approach, while the % of sessions/transactions precisely retrieved by any of the FPs obtained by *Apriori* is less than or equal to the one retrieved by any of the LET-FPs obtained by our approach. Furthermore, the LET-FPs obtained by our proposed method come with no pre-specified, fixed support, and require roughly half the time of *APriori*. *Zooming* results in increasing coverage because it can dig out *small localized LET-FPs*. Yet *precision is not significantly affected*. Finally, *Robustified* LET-FPs have the best coverage performance, followed by aggregated, and finally by the raw (not post-processed) LET-FPs.

4 Conclusions

In this paper, we applied a novel FP mining approach to mining *local, error-tolerant* frequent patterns (profiles) from Web transaction data. We also explored zooming as an efficient search strategy, studied several post-processing options of the mined frequent patterns, and investigated an information retrieval inspired validation procedure. Other important issues such as scalability will be addressed in the future.

Acknowledgment

This work is supported by National Science Foundation CAREER Award IIS-0133948 to O. Nasraoui.

References

1. C. Aggarwal, C. Procopiuc, and P. Yu. Finding Localized associations in market basket data. IEEE Trans. Kowledge and Data Engineering, Vol 14, No. 1, Jan 2002.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th Int'l Conf. on Very Large Databases*, SanTiago, Chile, June 1994.
3. D. Cheung, J. Han, V. Ng, and C. Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proc. of the 12__ Intl. Conf. on Data Engineering*, February 1996.
4. V. Ganti, J. Gehrke, and R. Ramakrishnan. Demon: Mining and monitoring evolving data. In *Proc. of the 16__ Int'l Conf. on Data Engineering*, pp. 439–448, May 2000.
5. J. Han, H. Jamil, Y. Lu, L. Chen, Y. Liao, and J. Pei. Dna-miner: A system prototype for mining dna sequences. In *Proc. of the 2001 ACM-SIGMOD Int'l. Conf. on Management of Data*, Santa Barbara, CA, May 2001.
6. C. Kamath. On mining scientific datasets. In et al R. L. Grossman, editor, *Data Mining for Scientific and Engineering Applications*, pages 1–21. Kluwer Academic Publishers, 2001.
7. O. Nasraoui and R. Krishnapuram, and A. Joshi. Mining Web Access Logs Using a Relational Clustering Algorithm Based on a Robust Estimator, 8th International World Wide Web Conference, Toronto, pp. 40-41, 1999.

8. J. Pei, A.K.H. Tung, and J. Han, Fault tolerant frequent pattern mining: Problems and challenges, Proc. 2001 ACM-SIGMOD Int. Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'01), Santa Barbara, CA, May 2001.
9. M. Rajman and R. Besan. Text mining - knowledge extraction from unstructured textual data. In *Proc. of the Int'l Conf. Federation of Classication Societies*, pages 473–480, Roma, Italy, 1998.
10. S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka. An efficient algorithm for the incremental updation of association rules. In *Proc. of the 3__ Int'l Conf. on Knowledge Discovery and Data Mining*, August 1997.
11. A. Veloso,W. Meira Jr., M. B. de Carvalho, B. Possas, S. Parthasarathy, and M. Zaki. Mining frequent itemsets in evolving databases. In *Proc. of the 2__ SIAM Int'l Conf. on Data Mining*, Arlington, USA, May 2002.
12. C. Yang, U. Fayyad, and P. Bradley, Efficient Discovery of error-tolerant frequent itemsets in high dimensions, In Proc. of seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 194-203, San Francisco, California, Aug. 2001.
13. M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New parallel algorithms for fast discovery of association rules. *Data Mining and Knowledge Discovery: An International Journal*, 4(1):343–373, December 1997.
14. O. Nasraoui and S. Goswami, Mining and Validating Localized Frequent Itemsets with Dynamic Tolerance, in Proceedings of SIAM conference on Data Mining (SDM 2006), Bethesda, Maryland, Apr 2006.